UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/729,771	12/08/2003	Erik de Groot	I20 04992US	3404

128 7590 02/06/2007

HONEYWELL INTERNATIONAL INC.
101 COLUMBIA ROAD
P O BOX 2245
MORRISTOWN, NJ 07962-2245

EXAMINER	
PHAM, THAI V	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/06/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/729,771	DE GROOT ET AL.	
	Examiner	Art Unit	
	Thai Van Pham	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 November 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Remarks

1. Applicant's amendment dated on 11/06/2006 responds to the Office Action dated on 08/02/2006, provided in the rejection of claims 1 – 37.

Claims 11, 15, and 27 have been amended

The specification, specifically paragraph [0055], has been sufficiently amended to correct typographical errors to overcome the previous objections.

Claims 15 and 27 have been sufficiently amended to correct typographical errors to overcome the previous objections.

Claims 1 – 37 remain pending in this application and have been fully considered by Examiner.

Note: Applicant's amendment does not comply with R 1.121(C). In principle of compact prosecution, Examiner assumes that "(AMENDED)" claims would be properly reworded as "(CURRENTLY AMENDED)" claims.

2. Applicant has amended the claims 11, 15, and 27 to recite:

-- Claim 11.

"*...providing a permission status to perform or not perform said action with said object.*"

-- Claim 15:

"*A computer readable medium having executable instructions stored thereon to perform a method of validating a state transition of a life cycle process in a source control system, said method comprising:*

determining whether a current state transition in a state transition request for an object from a user requires an electronic signature based on user-defined transition restrictions of said life cycle process ..."

-- Claim 27.

"*The system according to claim 26, further comprising:*

another processor to back-up said processor."

3. Examiner has fully considered Applicant's arguments with respect to the 35 USC § 102 and 103 rejections in the previous Office Action and concluded that they are unpersuasive as will be addressed under ***Prior Art's Arguments – Rejections*** section below.

4. The rejection of all claims over prior arts of record in the previous Office Action is maintained in light of the amendments to the claims and necessitated additional clarifications provided in this Office Action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP §706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Prior Art's Arguments – Rejection

5. Applicant's argument filed on 11/06/2006, in particular on pages 10 – 20, has been fully considered by Examiner.

-- Claim 1.

6. Applicant argues that CORBA (**Fiszman** Col.6: lines 20 – 50) does not describe: the change state function verifies "*compliance with said user-defined state transitions*" (Remarks, page 12: 1st paragraph).

7. Examiner's response:

Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Furthermore, Applicant is reminded and advised to read the prior art as a whole.

Fiszman discloses The GPAE engine that requests the application and components on a processing node to execute based attribute values of activities (Col. 6: lines 43 – 49). The transitional conditions between activities, i.e. states, in a process disclosed in **Fiszman** as depicted in Fig. 9 are *change state functions* that comply with *user-defined state transitional*

conditions in Fig. 12. Therefore, **Fiszman** discloses the step of: "*providing a change state function for changing a current state associated with an object to a next state associated with said object, said change state function verifying compliance with said user-defined state transitions*".

8. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 1.

-- Claim 11.

9. Applicant argues that **Fiszman** (Col. 5: lines 21 – 28) does not disclose: "*receiving a request to perform an action with said object*" since it generally describes of a process definition, not an object, not a state of an object and not a receipt of a request to perform an action with the object (Remarks, page 12: 2nd paragraph).

10. Examiner's response:

Fiszman discloses that a process definition involves human interaction (Col. 5: lines 21 – 28). The process definition is further depicted in details in Fig. 9 in which the user defines the parameters and conditions associated with activities in the process. In other words, the GUI receives specification, i.e. request, to determine transitions i.e. actions, of activities, i.e. object. Therefore, **Fiszman** discloses the step of "*receiving a request to perform an action with said object*".

11. Applicant argues that **Fiszman** (Figs. 11 and 17 and associated text, e.g. Col. 15: lines 30 – 37) does not disclose "*determining whether said object has ever been checked-in to a source*

control system" since it generally describes user selection of an activity from a list of available activities, but does not describe an object or a determination of whether the object has been checked into a source control system (Remarks, page 12: 3rd paragraph).

12. Examiner's response:

Fiszman specifically discloses a source control associated with the process automation system (Fig. 17 – "Source Control"). The activities of a process (as those shown in **Fiszman**, Fig. 11) are, thus, managed by the source control. It is understood by on of ordinary skill in the art that these activities are displayed in a "Browse" window only when they already have been determined to exist as existing components, i.e. checked in, of a process in a source control environment. Therefore, **Fiszman** discloses the step of "*determining whether said object has ever been checked-in to a source control system*".

13. Applicant argues that **Fiszman** (Figs. 11 and 17 and associated text, e.g., Col. 15: lines 30 – 37) does not disclose: "*whether said object is currently checked-in*" since it generally describes user selection of an activity from a list of available activities, but does not describe an object or a determination of whether the object is currently checked-in (Remarks, page 12: 4th paragraph).

14. Examiner's response:

See paragraphs 11 and 12 above.

15. Applicant argues that **Fiszman** (Fig. 11 and associated text, e.g. Col. 15: lines 30 – 37) does not disclose: "*retrieving a definition of said state of said object*" since it generally describes

user selection of an activity from a list of available activities, but does not describe an object, retrieval of the object, or a definition of a state of the object (Remarks, page 12: 5th paragraph).

16. Examiner's response:

Fiszman discloses that a particular activity and implicitly the attributes associated with it, i.e. definition of an activity, can be retrieved from a list of activities in "Browse" window (Fig. 11). Therefore, **Fiszman** discloses the step of "*retrieving a definition of said state of said object*". Further, **Fiszman**'s activities are objects because they are independent entities which can be stored as objects in source code or intermediate code.

17. Applicant argues that **Fiszman** (Fig. 13 and associated text, e.g. Col. 16: line 42 – Col. 17: line 8) does not disclose "*determining from said definition whether said action is permissible in said state*" since it generally describes the creation and amendment of an activity definition, but does not describe a determination from the definition "*of said state of said object*" whether the action is permissible in the state of the object (Remarks, page 13: 1st paragraph).

18. Examiner's response:

Fiszman discloses that activity definition attributes such as pre-conditions and post-conditions are retrieved and can be modified/added using "Activity Definition" window (Fig. 13). The pre-conditions and post-conditions must be proper with respect to the process as a whole. Therefore, **Fiszman** discloses the step of "determining from said definition whether said action is permissible in said state ".

19. Applicant has amended independent claim 11 to recite: "*providing a permission status to perform or not perform said action with said object*". Applicant further argues that **Fiszman**

(Fig. 1 and associated text, e.g. Col. 5, lines 35 – 40) does not disclose this step since it generally describes the running of a process in response to a request that identifies the process and a specific instance of the process. There is no description of a permission status "*to perform or not perform said action with said object*" (Remarks, page 13: 2nd paragraph).

20. Examiner's response:

Fiszman also discloses that the feedback and output of GPAE provided as a result of its compilation as well as execution (Fig. 1 and associated text, e.g. Col. 5, lines 35 – 40). The transition properties from one state to another, i.e. action, are modified and subsequently applied (selecting "Apply"), i.e. perform, or not applied (select "Cancel"), i.e. not perform, to the process under execution (Fig. 12 and associated text, e.g. Col. 15: line 38 – Col. 16: line 41). Therefore, **Fiszman** discloses the step of "*providing a permission status to perform or not perform said action with said object*".

21. As a result, Applicant's amendment to Claim 11 and argument are found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 11.

-- Claim 13:

22. Applicant argues that **Fiszman** (Fig. 16b and associated text, e.g. Col. 17: line 59 – Col. 18: line 36) does not disclose the step of "*determining whether a next state in a state transition request from a user is allowed from a current state in said state transition request based on user-defined transition restrictions*" since it describes a breakdown of the store process of Fig. 16a. There is no description of a "*next state transition request from a user*", a determination of

allowability of a "*request*" that is "*based on user-defined transition restrictions*" (Remarks, page 13: 3rd paragraph).

23. Examiner's response:

Fiszman discloses that a process definition, such as the "store process" of Fig. 16a as portrayed in detail in Fig. 16b, is the a result of user inputs, i.e. requests from a user, in Figs. 9 – 15. The pre-conditions and post-conditions are correlated with the transitional conditions from one state to the next by in the process definition so that the transition is allowed only when all these conditions are satisfied. Therefore, **Fiszman** discloses the step of "*determining whether a next state in a state transition request from a user is allowed from a current state in said state transition request based on user-defined transition restrictions*".

24. Applicant argues that **Fiszman** (Fig. 16b and associated text, e.g. Col. 17: line 59 – Col. 18: line 36) does not disclose the step of "*determining whether said user has permission to make said state transition based on user-defined transition restrictions*" since it describes a breakdown of the store process of Fig. 16a. There is no description of the recited determination (Remarks, page 13: 4th paragraph).

25. Examiner's response:

Fiszman discloses that "access control privileges", i.e. permission for user, (Col. 5: line 42 – Col. 6: line 3) is required for GPAE and in order to create a process definition (Figs. 11 and 16b and associated text). If a user does not have the proper "access control privilege", he would not have access control for the process execution. Therefore, **Fiszman** discloses the step of "*determining whether said user has permission to make said state transition based on user-defined transition restrictions*".

26. Applicant argues that **Fiszman** (Fig. 1 and associated text, e.g. Col. 5: lines 35 – 40) does not disclose the step of "*providing a state transition status*" since it generally describes the running of a process in response to a request that identifies the process and a specific instance of the process. There is no description of "providing a state transition status" (Remarks, page 14: 1st paragraph).

27. Examiner's response:

Fiszman discloses that the feedback and output of GPAE provided as a result of its compilation as well as execution (Fig. 1 and associated text, e.g. Col. 5, lines 35 – 40). The feedback is well understood by one of ordinary skill in the art to include error, success, and warning information of the process compilation as well as execution. Therefore, **Fiszman** discloses the step of "*providing a state transition status*".

28. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 13.

-- Claim 16:

29. Applicant argues that **Fiszman** (Fig. 17 and associated text, e.g. Col. 18: line 37 – Col. 19: line 4) does not disclose "*determining whether said object is being checked-in for a first time*" since describes a source control repository that stores source code under version control (without any further description of version control). Applicant challenges Examiner's assertion of this inherent property of version control (Remarks, page 14: 2nd paragraph)

30. Examiner's response:

It is well understood by one of ordinary skill in the art that source-control software manages file versioning by successively incrementing the version number of the file as it is modified and checked in to the control software. If the file is checked-in for the first time, the software assigns #1 and/or similar notations to the file's properties to identify that this is the first version of the file. Examiner's inherency assertion in the previous rejection with regards to this property of a source control system should be well understood and clear to one of ordinary skill in the art of software development. If Applicant believes these are not inherent components and properties of a personal computer, Applicant must successfully traverse Examiner's assertion of these inherency arguments. Please see MPEP § 2112 (V): Once a reference teaching product appearing to be substantially identical is made the basis of a rejection, and the Examiner presents evidence or reasoning tending to show inherency, the burden shifts to the Applicant to show an unobvious difference.

31. Applicant argues that **Fiszman** (Fig. 9 and associated text, e.g. Col. 14: line 43 – Col. 15: line 13) does not disclose the step of "*retrieving a first fallback state for a first pre-defined state, if said object is being checked-in for said first time*" since it describes the creation of a process definition, but does not describe "*a first fallback state*", an "*object*" being "*checked-in for the first time*". **Fiszman's** defined activities are not objects (Remarks, page 14: 3rd paragraph).

32. Examiner's response:

Fiszman's defined activities are objects (see paragraphs 10 and 16 above).

Fiszman discloses "*a first fallback state*" for an activity, a process, and a transition (Fig. 9 and associated text, e.g. Col. 14: line 43 – Col. 15: line 13) as a circle, square, and arrow, respectively. The basic properties of the activity, process, and transition are assigned by default

in the circle, square, and arrow when they are first created, i.e. checked-in for the first time; while specific properties are modified and/or added as attributes and pre-conditions and post-conditions. Therefore, **Fiszman** discloses the step of "*retrieving a first fallback state for a first pre-defined state, if said object is being checked-in for said first time*".

33. Applicant argues that **Fiszman** (Fig. 9 and associated text, e.g. Col. 14: line 43 – Col. 15: line 13) does not disclose the step of "*providing said first fallback state, if said object is being checked-in for said first time*" since it describes the creation of a process definition, but does not describe "*a first fallback state*", an "*object*" being "*checked-in for the first time*". **Fiszman's** defined activities are not objects (Remarks, page 14: 4th paragraph).

34. Examiner's response:

See paragraphs 32 above.

35. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 16.

-- Claim 17:

36. Applicant argues that **Fiszman** (Figs. 13 and 14 and associated text, e.g. Col. 16: line 42 – Col. 17: line 8) does not disclose the step of "*providing said current fallback state, if said object is not being checked-in for said first time*" since it describes the creation and amendment of an activity definition (Remarks, page 15: 2nd paragraph).

37. Examiner's response:

See paragraphs 32 above.

38. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 17.

-- Claim 18:

39. Applicant argues that **Fiszman** (Figs. 9 – 11 and associated text, e.g. Col. 14: line 43 – Col. 15: line 37) does not disclose the step of "*receiving a definition of a new state from a user, said definition including a name and a fallback state*" since it describes the creation and amendment of an activity definition but does not describe "*receiving a definition of a new state from a user*". **Fiszman**'s user selects states from a plurality of predefined states (Remarks, page 15: 3rd paragraph).

40. Examiner's response:

See paragraphs 32 above. Activities and processes, i.e. states, are created in Fig. 9 using a fallback state, e.g. circle or rectangle, and associating a new state name, e.g. Upload and Notification. Specific attributes of the state are then modified and/or added as shown in Fig. 13 and thereby, defining a new and unique state. Therefore, **Fiszman** discloses the step of "*receiving a definition of a new state from a user, said definition including a name and a fallback state*".

41. Applicant argues that **Fiszman** (Figs. 9 – 11 and associated text, e.g. Col. 14: line 43 – Col. 15: line 37) does not disclose the step of "*determining whether said name is unique among existing state definitions*" since it generally describes the creation and amendment of an activity

definition, but does not describe "*receiving a definition of a new state from a user*". **Fiszman**'s user selects states from a plurality of predefined states (Remarks, page 15: 4th paragraph).

42. Examiner's response:

See paragraph 40 above.

43. Applicant argues that **Fiszman** (Fig. 4 and associated text) does not disclose the step of "*validating said fallback state* " (Remarks, page 15: 5th paragraph).

44. Examiner's response:

See paragraph 32 above for **Fiszman**'s "fallback states".

Fiszman discloses that user-defined processes and activities and validated at the "Process Instance Server" that "Enacts Activity, Provides Transaction Control, Updates Status" (Fig. 4 and associated text). The fallback state are thus implicitly validated as its associated activity or process is validated. Therefore, , **Fiszman** discloses the step of "*validating said fallback state* ".

45. Applicant argues that **Fiszman** (Fig. 17 and associated text) does not discloses the step of "*adding said definition to a source control system, only if said name is unique and said fallback state is valid*" since this is not an inherent property of version control (Remarks, page 16: 1st paragraph).

46. Examiner's response:

See paragraph 30 for the **Fiszman**'s use of source control in his invention.

Once **Fiszman**'s processes and activities are newly defined (Fig. 9) and validated (Fig. 4), they are added to a source control (Fig. 17) for versioning and release management as well understood by one of ordinary skill in the art. The names of the process/activity must be unique

in names, i.e. said name is unique, and must be compliant to the predetermined conditions, i.e. said fallback state is valid, in order for them to be a newly added process/activity of in the source control. This is an inherent property of any source control system. Therefore, **Fiszman** discloses the step of "*adding said definition to a source control system, only if said name is unique and said fallback state is valid*".

47. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 18.

-- Claim 21:

48. Applicant argues that **Fiszman** (Figs. 9 – 11 and associated text, e.g. Col. 14: line 43 – Col. 15: line 37) does not disclose the step of "*receiving a modified definition of a state from a user, said modified definition including a name and a fallback state*" since it generally describes the creation and amendment of an activity definition, but does not describe "*receiving a modified definition of a state from a user*". **Fiszman**'s user selects states from a plurality of predefined states (Remarks, page 16: 2nd paragraph).

49. Examiner's response:

See paragraph 40 above.

50. Applicant argues that **Fiszman** (Figs. 9 – 11 and associated text, e.g. Col. 14: line 43 – Col. 15: line 37) does not disclose the step of "*determining whether said name is unique among existing state definitions*" since it describes the creation and amendment of an activity definition,

but does not describe "*receiving a definition of a new state from a user*". **Fiszman**'s user selects states from a plurality of predefined states (Remarks, page 16: 3rd paragraph).

51. Examiner's response:

See paragraph 40 above.

52. Applicant argues that **Fiszman** (Fig. 4 and associated text) does not disclose the step of "*validating said fallback state*" (Remarks, page 16: 4th paragraph).

53. Examiner's response:

See paragraph 44 above.

54. Applicant argues that **Fiszman** (Fig. 17 and associated text) does not discloses the step of "*updating said modified definition in a source control system, only if said name is unique and said fallback state is valid*" since this is not an inherent property of version control (Remarks, page 16: 5th paragraph).

55. Examiner's response:

See paragraph 46 above.

56. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 21.

-- Claim 26:

57. Applicant argues that **Fiszman** (Col. 7: line 49 – Col. 8: line 6) does not disclose limitation of "*a controller in communication with said processor via a network to be loaded with said objects to provide process control for a plurality of devices*" since it describes a model view

controller for GUI displays to users and not a controller that is "*loaded with said objects to provide process control for a plurality of devices*" (Remarks, page 17: 2nd paragraph).

58. Examiner's response:

Fiszman discloses the GPAE (Figs. 3 and 4 and associated text) that implements "MVC pattern between the work item definitions/runtime entities (model), the controller (the GPAE's APIs) and the GUI (view) and ... " (Col. 7: line 49 – Col. 8: line 6). GPAE is also interfaced with CORBA. The controller here is the GPAE's APIs which must be in communication with one or more processors via a network, e.g. CORBA, to be loaded with work item definitions/runtime entities, i.e. objects. Therefore, **Fiszman** discloses the limitation of "*a controller in communication with said processor via a network to be loaded with said objects to provide process control for a plurality of devices*".

59. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 21. Consequently, the rejection of claims 1 – 3, 7, 8, 11, 13, 16 – 18, 20, 21,23, 26, 28 – 30, 32, 33, 36 and 37 under 35 U.S.C. 102(b) in the previous Office Action as anticipated by **Fiszman** is maintained.

-- Claims 24 and 25:

60. Applicant argues that deletion of a state definition is unobvious over **Fiszman**'s disclosure (Figs. 9 – 12) (Remarks, page 17: 5th paragraph).

61. Examiner's response:

In the rejection of claims 24 and 24 in the previous Office Action, Examiner has established a prima facie case of obviousness. It is well known to one of ordinary skill in the art

that although **Fiszman** describes adding and selecting activities and process in GPAE (Fig. 9 and associated text), deleting a state must have been provided in this process of graph creation when an activity/process is mistakenly created or obsolete by modification of the activity/process flow. The concept of adding and modifying a state being accompanied by deleting of it is very much analogous that seen in any graphical flow chart toolbox, such as Microsoft Visio or Microsoft Word.

Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. If Applicant truly believes that deletion of a state definition is unobvious over **Fiszman**'s disclosure, Applicant must adequately traverse the rejection as well as provide evidence to support the argument. Please See MPEP § 2144.03 [C]: "...To adequately traverse such a finding, an applicant must specifically point out the supposed errors in the examiner's action, which would include stating why the noticed fact is not considered to be common knowledge or well-known in the art. See 37 CFR 1.111(b). See also Chevenard, 139 F.2d at 713, 60 USPQ at 241 ("[I]n the absence of any demand by appellant for the examiner to produce authority for his statement, we will not consider this contention."). A general allegation that the claims define a patentable invention without any reference to the examiner's assertion of official notice would be inadequate ... ".

62. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claims 24 and 25.

-- Claim 27:

63. Applicant challenges Examiner's Official Notice on "*another processor to back-up said processor*". Moreover, Applicant argues that dependent claim 27 is unobvious over **Fiszman** because **Fiszman** lacks the controller recited in independent claim 26 for the reasons set forth in the discussion of independent claim 26 (Remarks, page 18: 2nd paragraph).

64. Examiner's response:

In the rejection of claim 27 in the previous Office Action, Examiner has established a prima facie case of obviousness. It is well known to one of ordinary skill in the art that crucial project data on corporate computers are periodically backed up by local and/or central servers for record retrieval when necessary. Each server inherently consists of one or more processors. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. If Applicant truly believes that the limitation recited in this claim is unobvious over **Fiszman**'s disclosure, Applicant must adequately traverse the rejection as well as provide evidence to support the argument. Please See MPEP § 2144.03 [C]: "...To adequately traverse such a finding, an applicant must specifically point out the supposed errors in the examiner's action, which would include stating why the noticed fact is not considered to be common knowledge or well-known in the art. See 37 CFR 1.111(b). See also Chevenard, 139 F.2d at 713, 60 USPQ at 241 ("[I]n the absence of any demand by appellant for the examiner to produce authority for his statement, we will not consider this contention."). A general allegation that the claims define a patentable

invention without any reference to the examiner's assertion of official notice would be inadequate ... ".

65. As a result, Applicant's argument is found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of claim 27.

-- Claim 15:

66. Applicant has amended independent claim 15 to recite: "*a method of validating a state transition of a life cycle process in a source control system*" and "*an electronic signature based on use-defined transition restrictions of said life cycle process*". As a result of the amendment, Applicant argues that **Wisnosky** does not disclose a life cycle process. Therefore, amended independent claim 15 is unobvious in view of **Wisnosky**. Applicant further argues that **Wisnosky** is an improper reference because it is for a non-analogous art. Amended independent claim 15 is for a method of validating a state transition of a life cycle process in a source control system, whereas **Wisnosky** is directed to the art of "measuring educational performance levels". Therefore, Applicant argues that these two arts are unrelated and non-analogous (Remarks, page 18: 5th and 6th paragraphs).

67. Examiner's response:

In response to applicant's argument that **Wisnosky** is nonanalogous art, it has been held that a prior art reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the applicant was concerned, in order to be relied upon as a basis for rejection of the claimed invention. See In re Oetiker, 977 F.2d

1443, 24 USPQ2d 1443 (Fed. Cir. 1992). In this case, both Applicant and **Wisnosky**'s disclosures are directed to using software automation tool to manage processes; and this factor alone indicates that the two disclosures belong to analogous arts. The differences between what processes the software are being applied to is irrelevant in the determination of analogous art. Furthermore, Applicant's disclosure of validation of a state transition of a life cycle process in a source control system is analogous to Wisnosky's disclosure of validation of an activity or process transition of a workflow management process in a source control system.

Wisnosky discloses that a method of validating a state transition of a workflow process (Abstract and Fig. 4 and associated text), i.e. a life cycle process, in a source control system (Fig. 17 and associated text). Examiner maintains the Official Notice taken in the rejection of the previous Office Action with regard to the limitation recited in claim 15: "*an electronic signature based on use-defined transition restrictions of said life cycle process*".

68. As a result, Applicant's amendment to Claim 15 and argument are found unpersuasive to overcome the previous prior art rejection; thus, Examiner maintains the previous rejection of Claim 15. Consequently, the rejection of claims 4 – 6, 9, 10, 12, 14, 19, 22, 31, 34 and 35 under 35 U.S.C 103(a) as unpatentable over **Fiszman** in view of **Wisnosky** are maintained.

69. Applicant argues that Examiner's previous Office Action suggestion to combine Fiszman with Wisnosky is improperly based on the hindsight of Applicants' disclosure. Such hindsight reconstruction of the art cannot be the basis of a rejection under 35 U.S.C. 103. The prior art itself must suggest that modification or provide the reason or motivation for making such modification. In re Laskowski, 871 F.2d 115, 117, 10 USPQ 2d 1397, 1398-1399 (CAFC, 1989).

"The invention must be viewed not after the blueprint has been drawn by the inventor, but as it would have been perceived in the state of the art that existed at the time the invention was made." Sensonics Inc. v. Aerosonic Corp. 38 USPQ 2d 1551, 1554 (CAFC, 1996), citing Interconnect Plannin.q Corp. v. Feil, 774 F. 2d 1132, 1138, 227 USPQ 543,547 (CAFC, 1985) (Remarks, page 19: 4th paragraph).

70. Examiner's response:

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See In re McLaughlin, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Claim Rejections

71. Claims 1 – 37 stand finally rejected as based on arts of record.

Claims 1 – 3, 7 – 8, 11, 13, 16 – 18, 20 – 21, 23, 26, 28 – 30, 32 – 33, and 36 – 37 are rejected under 35 U.S.C. 102(b) as being anticipated by **Fiszman** (6,115,646).

Claims 24 – 25, and 27 are rejected under 35 U.S.C. 103(a) as being obvious over **Fiszman** (6,115,646).

Claim 15 is rejected under 35 U.S.C. 103(a) as being obvious over **Wisnosky** (2003/0190593).

Claims 4 – 6, 9 – 10, 12, 14, 19, 22, 31, and 34 – 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Fiszman** (6,115,646) and in view of **Wisnosky** (2003/0190593).

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

72. Claims 1 – 3, 7 – 8, 11, 13, 16 – 18, 20 – 21, 23, 26, 28 – 30, 32 – 33, and 36 – 37 are rejected under 35 U.S.C. 102(b) as being anticipated by **Fiszman** (6,115,646).

-- Claim 1: **Fiszman** discloses a method for enforcing a life cycle process in a source control system, comprising:

- receiving a user-defined life cycle process having a plurality of states, each state having attributes (i.e., activities and their individual attributes; Fig. 1, page 5 line 7 – page 6 line 3);
- receiving user-defined state transitions between said plurality of states (i.e., transition properties of activities; Fig. 12, page 15 line 38 – page 16 line 41);
- providing a change state function for changing a current state associated with an object to a next state associated with said object, said change state function verifying compliance with said user-defined state transitions (page 6 lines 20 – 50);
- and providing version control for said object in said source control system (Fig. 17, page 18 lines 44 – 63).

-- Claim 2: **Fiszman** discloses the method according to claim 1 where the version control inherently must comprise providing a check-in function and providing a check-out function.

-- Claim 3: **Fiszman** discloses the method according to claim 1 and further discloses the attributes include a fallback state (i.e., activities defined within a particular process; Fig. 9, page 14 line 43 – page 15 line 13).

-- Claim 7: **Fiszman** discloses the method according to claim 1, where the object is a control strategy for a process control system (page 6 lines 30 – 34).

-- Claim 8: **Fiszman** discloses the method according to claim 7, where the attributes include whether said control strategy is loadable to a controller (page 7 line 49 – page 8 line 6).

-- Claim 11: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of determining permissions for actions with an object based on a state of said object, said method comprising:

- receiving a request to perform an action with said object (i.e., execution of an a particular activity by a processing node or a role; page 5 lines 21 – 28);
- determining whether said object has ever been checked-in to a source control system (i.e., determining existence of activity associated with version control; Figs. 11 and 17, page 15 lines 30 –37);
- determining whether said object is currently checked-in (i.e., determining existence of activity associated with version control; Figs. 11 and 17, page 15 lines 30 –37);

• retrieving a definition of said state of said object (i.e., displaying existing activities definitions; Fig. 11, page 15 lines 30 –37);

• determining from said definition whether said action is permissible in said state (i.e., attributes defined for an activity; Fig. 13, page 16 line 42 – page 17 line 8); and

• providing a permission status to perform or not perform said action with said object (i.e., feedback and output; Fig. 1, page 5 lines 35 – 40).

-- Claim 13: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of validating a state transition, said method comprising:

• determining whether a next state in a state transition request from a user is allowed from a current state in said state transition request based on user-defined transition restrictions (Fig. 16B, page 17 line 59 – page 18 line 36);

• determining whether said user has permission to make said state transition based on user-defined transition restrictions (Fig. 16B, page 17 line 59 – page 18 line 36); and

• providing a state transition status (Fig. 1, page 5 lines 35 – 40).

-- Claim 16: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of determining a new state for an object version upon check-in, said method comprising:

• determining whether said object is being checked-in for a first time (i.e., an inherent property of version control; Fig. 17);

• retrieving a first fallback state for a first pre-defined state, if said object is being checked-in for said first time (i.e., activities defined within a particular process; Fig. 9, page 14 line 43 – page 15 line 13); and

• providing said first fallback state, if said object is being checked-in for said first time (i.e., activities defined within a particular process; Fig. 9, page 14 line 43 – page 15 line 13).

-- Claim 17: **Fiszman** discloses the computer readable medium according to claim 16, comprising:

• retrieving a current state for a current version of said object, if said object is not being checked-in for said first time (i.e., retrieving an existing activity definition; Figs. 13 –14, page 16 line 42 – page 17 line 8);

• retrieving a current fallback state for said current state of said object, if said object is not being checked-in for said first time (i.e., retrieving an existing activity definition; Figs. 13 – 14, page 16 line 42 – page 17 line 8); and

• providing said current fallback state, if said object is not being checked-in for said first time (i.e., retrieving an existing activity definition; Figs. 13 –14, page 16 line 42 – page 17 line 8).

-- Claim 18: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of processing the addition of a state, said method comprising:

• receiving a definition of a new state from a user, said definition including a name and a fallback state (i.e., creating activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37);

• determining whether said name is unique among existing state definitions (i.e., listing activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37);

• validating said fallback state (i.e., distinctive activities; Fig. 4); and

• adding said definition to a source control system, only if said name is unique and said fallback state is valid (i.e., an inherent property of version control; Fig. 17).

-- Claim 20: **Fiszman** discloses the computer readable medium according to claim 18 and further discloses that the method further comprising determining whether said user has a privilege to edit said definition; and wherein said adding said definition to said source control system is performed on an additional condition of whether said user has said privilege (i.e., access authorization; page 5 lines 53 – 56).

-- Claim 21: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of processing the modification of a state, said method comprising:

• receiving a modified definition of a state from a user, said modified definition including a name and a fallback state (i.e., editing activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37);

• determining whether said name is unique among existing state definitions (i.e., listing activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37);

• validating said fallback state (i.e., distinctive activities; Fig. 4); and

• updating said modified definition in a source control system, only if said name is unique and said fallback state is valid (i.e., an inherent property of version control; Fig. 17).

-- Claim 23: **Fiszman** discloses the computer readable medium according to claim 21, further comprising: determining whether said user has a privilege to edit said definition; and wherein said updating said modified definition in said source control system is performed on an additional condition of whether said user has said privilege (i.e., access authorization; page 5 lines 53 – 56).

-- Claim 26: **Fiszman** discloses a source control system for a process control system, comprising:

• a processor (an inherent property in a computer system);

• a life cycle process component executable on said processor to enforce compliance with user-defined life cycle states (Fig. 3, page 8 lines 11 – 33);

• a version control component executable on said processor to associate a version number with each object (Fig. 17, page 18 lines 37 – 63); and

• a controller in communication with said processor via a network to be loaded with said objects to provide process control for a plurality of devices (page 7 line 49 – page 8 line 6).

-- Claim 28: **Fiszman** discloses the system according to claim 26, further comprising: a state configuration component executable on said processor to receive state information from a user for each state (i.e., defining activity attributes; Figs. 9 –13).

-- Claim 29: **Fiszman** discloses the system according to claim 28, wherein said state information includes a state name and an indication of whether load to controller is allowed from that state (Figs. 13 and 14, page 16 line 42 – page 15 line 37; page 7 line 49 – page 8 line 6).

-- Claim 30: **Fiszman** discloses the system according to claim 28, wherein said state information includes a fallback state (i.e., activities defined within a particular process; Fig. 9, page 14 line 43 – page 15 line 13).

-- Claim 32: **Fiszman** discloses the system according to claim 28, wherein said state configuration component provides editing functions for said state information (Fig. 13, page 16 lines 42 – 45).

-- Claim 33: **Fiszman** discloses the system according to claim 26, further comprising: a state transition component executable on said processor to receive state transition configuration requirements from a user (Fig. 12, page 15 line 38 – page 16 line 41).

-- Claim 36: **Fiszman** discloses the system according to claim 26, wherein said version control component provides check-in and check-out functions (i.e., an inherent property of version control; Fig. 17).

-- Claim 37: **Fiszman** discloses the system according to claim 26, further comprising: a change qualification state component to process a qualification state transition request from a user (Fig. 12, page 15 line 38 – page 16 line 41).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

73. Claims 24 – 25, and 27 are rejected under 35 U.S.C. 103(a) as being obvious over **Fiszman** (6,115,646).

-- Claim 24: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of processing the addition and modification of a state, the method comprising:

• receiving a modified definition of a state from a user, said modified definition including a name and a fallback state (i.e., editing activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37). Although **Fiszman** does not explicitly disclose that the method comprises receiving a request to delete a state definition for said state from a user, **Fiszman**'s disclosure of the modified definition of an activity above would have made it obvious to one of

ordinary skills in the art of software development at the time the invention was made to add to the method of **Fiszman** a method of processing a deletion of an activity when the activity becomes obsolete in a process or when it is mistakenly created by the user. The deletion of the activity can be performed in the activity graph of the process definition in Fig. 9 where the addition and modification take place;

• determining whether said name is unique among existing state definitions and validating said fallback state (i.e., listing activities in a process definition; Figs. 9 – 11, page 14 line 43 – page 15 line 37). Although **Fiszman** but does not explicitly disclose that the method comprises determining whether said state definition is referenced by any other state definition in a source control system and determining whether any objects in said source control system have a current state equal to said state, **Fiszman**'s disclosure of defining/modifying the attributes of activities (Figs. 9 and 13) shows how transition properties of an activity are determined in relation to the existence of activities to and from which the transitions take place under certain conditions. Therefore, prior to an activity deletion, it is necessary to examine the dependencies of the activity to make proper to modification to the process as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further allow the method of processing the deletion of a state in **Fiszman** to determine whether the state is referenced by any other state in the source control system; and

• updating said modified definition in a source control system, only if said name is unique and said fallback state is valid (i.e., an inherent property of version control; Fig. 17). Although **Fiszman** does not further explicitly disclose that the method deleting said state definition from said source control system, only if said state definition is not referenced by any

other state definition in said source control system and no objects in said source control system have said current state equal to said state, it would have been obvious that once an activity of the method is determined to be obsolete and proper modification to the process is made to purge the process of any reference made to the obsolete activity, the activity can be removed safely. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further allow the method of processing the deletion of a state in **Fiszman** to delete the state definition when it is obsolete in the source control system.

-- Claim 25: **Fiszman** discloses the computer readable medium according to claim 24, further comprising: determining whether said user has a privilege to delete said definition; and wherein said deleting said state definition from said source control system is performed on an additional condition of whether said user has said privilege (i.e., access authorization; page 5 lines 53 – 56).

-- Claim 27: **Fiszman** discloses the system according to claim 26, further comprising: another processor to back-up said server. Official Notice is taken that it is old and well known in the art of software development that data on a computer can always be backed up over a network. Thus, the process control system can be backed up by a secondary server, which inherently must contain a processor. Therefore, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further allow the method of processing the deletion of a state in **Fiszman** to have a secondary back-up server containing a processor.

74. Claim 15 is rejected under 35 U.S.C. 103(a) as being obvious over **Wisnosky** (2003/0190593).

-- Claim 15: **Wisnosky** discloses a computer readable medium having executable instructions stored thereon to perform a method of validating a state transition of a life cycle process in a source control system, said method comprising:

• determining whether said current state transition in a state transition request for an object from a user requires an electronic signature based on user-defined transition restrictions of said life cycle process (i.e., different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system; Fig. 6D, [0081 – 0082]);

• but **Wisnosky** does not explicitly disclose that the method determines whether a previous state transition for said object required a previous electronic signature, if said current state transition requires a current electronic signature; allowing said current state transition only if said previous electronic signature is different than said current electronic signature. Official Notice is taken that it is old and well known in business accounting practices that the process for generating a purchase order always requires at least two different signatures for approval. In particular, a person who makes the determination of what needs to be purchased generates a purchase order explaining the need and detailing the price and specification of the items to be purchased; he then signs it to authenticate the order (i.e., purchase order generation state). Subsequently, a manager or a person having the authority to approve the purchase order must sign it (i.e., purchase order approval state) before a check is issued my another person having the authority to make the financial decision (i.e., payment state). It is clear in this example that the

signature required to go from "purchase order approval state" to "payment state" must be different than that required to go from "purchase order generation state" to "purchase order approval state". Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further allow the method of **Wisnosky** to compare previous and current electronic signatures to validate the current state transition;

- providing a validation status (Fig. 6D, [0081 – 0082]).

75. Claims 4 – 6, 9 – 10, 12, 14, 19, 22, 31, and 34 – 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Fiszman** (6,115,646) and in view of **Wisnosky** (2003/0190593).

-- Claim 4: **Fiszman** discloses the method according to claim 1 but does not explicitly disclose the method further comprising: receiving user-defined security for said user-defined state transitions. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 –

56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further provide the method of **Fiszman** with a user-defined security for state transitions in addition to access authorization.

-- Claim 5: **Fiszman** and **Wisnosky** disclose the method according to claim 4 and **Wisnosky** further discloses the user-defined security includes electronic signatures (Fig. 5, [0076]).

-- Claim 6: **Fiszman** and **Wisnosky** disclose the method according to claim 4 and **Wisnosky** further discloses the user-defined security includes which users have permission to make which state transitions (i.e., read – write – delete; Fig. 6D, [0081 – 0082]).

-- Claim 9: **Fiszman** and **Wisnosky** disclose the method according to claim 1 and **Wisnosky** further discloses that receiving said user-defined life cycle process having said plurality of states, each state having attributes is performed through a user interface having an editable table, said table having state names as rows and attributes as columns and having cells indicating values for said attributes (Fig. 6D, [0081 – 0082]).

-- Claim 10: : **Fiszman** and **Wisnosky** disclose the method according to claim 6 and **Wisnosky** further discloses that receiving user-defined state transitions between said plurality of states is performed through a user interface having an editable table, said table having state names as rows and column and having cells indicating which users have permission to make which state transitions (i.e., read – write – delete for each specific area; Fig. 6D, [0081 – 0082]).

-- Claim 12: **Fiszman** discloses a computer readable medium having executable instructions stored thereon to perform a method of validating state transitions, said method comprising:

- receiving a request to make a state transition for an object from a user (page 5 lines 21 – 28);
- determining whether said object is checked-in (an inherent property of version control; Fig. 11, page 15 lines 30 –37);
- however, **Fiszman** does not explicitly disclose that the method determines whether said user has permission to make said state transition based on a user-defined state transition model. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write

permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further provide the method of **Fiszman** with the ability to determe whether the user has permission to make said state transition based on a user-defined state transition model;

- permitting said state transition, if said user has permission. If a user satisfies the security requirements, it is obvious for the method to permit the state transition; and
- providing a state transition status (Fig. 1, page 5 lines 35 – 40).

-- Claim 14: **Fiszman** discloses the computer readable medium according to claim 13 but does not explicitly disclose the method further comprising determining whether said state transition has a restricted signing requirement and, if so, verifying that said restricted signing requirement is met. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and

furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further enable security for user-defined state transition model in the method of **Fiszman** , and furthermore, provide the model with the ability to determine if a state transition has a restricted signing requirement as well as to verify if the restricted signing requirement is met.

-- Claim 19: **Fiszman** discloses the computer readable medium according to claim 18 but does not explicitly disclose that the definition includes a restricted signing requirement and further comprising validating said restricted signing requirement; and wherein said adding said definition to said source control system is performed on an additional condition of whether said

restricted signing requirement is valid. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further include a restricted signing requirement in the state definition in the method of **Fiszman** , and furthermore, provide the model with the ability to validate the restricted signing requirement and to restrict adding the state definition to the source control system on an additional condition of whether said restricted signing requirement is valid.

-- Claim 22: **Fiszman** discloses the computer readable medium according to claim 21 but does not explicitly disclose that the definition includes a restricted signing requirement and further comprising: validating said restricted signing requirement; and wherein said updating said modified definition in said source control system is performed on an additional condition of whether said restricted signing requirement is valid. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of

ordinary skills in the art of software development at the time the invention was made to further include a restricted signing requirement in the state definition in the method of **Fiszman** , and furthermore, provide the model with the ability to validate the restricted signing requirement and to restrict updating the state definition to the source control system on an additional condition of whether said restricted signing requirement is valid.

-- Claim 31: **Fiszman** discloses the system according to claim 28 but does not explicitly disclose that the state information includes an indication of whether restricted signing is needed. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings

would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further add to the state definition of the method of **Fiszman** an optional restricted signing requirement which facilitates the state information to include an indication of whether restricted signing is needed.

-- Claim 34 and 35: **Fiszman** discloses the system according to claim 33 but does not explicitly disclose that the state transition configuration requirements include which users have permission to make particular state transitions as well as an indication of whether an electronic signature is needed to make particular state transitions. **Wisnosky** discloses a system implementing a method for automatically generating individual transition plans according to user input statistical data; the system includes security setting that permits access to certain areas of the plan by authorized users. Only authorized user is granted access to the system implementing the method of **Wisnosky** (Fig. 5, [0076]); and furthermore, different areas of the system employ various security settings to enable a certain user to gain different access levels to a particular area of the system (Fig. 6D, [0081 – 0082]). Specifically, a particular user group (e.g., managers) is given all read, write, and delete permissions to an enterprise area; whereas a different user group (e.g., workers) is given only read and write permissions to the same area. The states of an individual transition plan change as data is created or removed by a user with proper authorization. The system implementing the method of **Fiszman** requires only user authorization for gaining system access (page 5 lines 53 – 56); and it would be more secured and robust if it further requires

specific user security settings as disclosed **Wisnosky** so that only certain groups of users are given permissions to modify and/or delete certain processes or their corresponding activities. In other words, specific user security settings would provide the method of **Fiszman** with the ability to enable user-defined security for individual activity transitions and/or process transitions as a whole. Thus, it would have been obvious to one of ordinary skills in the art of software development at the time the invention was made to further add to the state transition configuration requirements of the method of **Fiszman** an optional restricted signing requirement which facilitates the state transition configuration requirements to include an indication of whether restricted signing is needed.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thai Van Pham whose telephone number is (571) 270-1064. The examiner can normally be reached on Monday - Thursday, 8am - 3pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TVP

TUAN DAM
SUPERVISORY PATENT EXAMINER